Honeywell

O16-XREF, SSUP, AND MAC SOURCE LANGUAGE PROCESSORS

SERIES 16

SOFTWARE

Honeywell

O16-XREF, SSUP, AND MAC SOURCE LANGUAGE PROCESSORS

SERIES 16

SUBJECT:

Description, System Generation, Operation, Memory Requirements, and Programming Examples of Three Source Language Processors.

DATE:

July 1971

ORDER NUMBER:

AC94, Rev. 0 (Formerly M-459)

DOCUMENT NUMBER:

70130072582A

PREFACE

This manual provides the programmers and operators of Series 16 computer systems with the information necessary to use the following Honeywell-supplied source language processors:

O16-XREF	Cross-Reference Listing
SSUP	Symbolic Source Update
MAC	Macro Instruction Processor

Elements of the languages, examples of their use, exception conditions, and related supporting programs are described. Instructions for generating stand-alone systems and instructions for executing the programs are also included.

The reader is assumed to have a basic familiarity with Series 16 assembly language programming and to have read the 316/516 Programmers' Reference Manual, Order Number BX47 (formerly M-490).

The illustration on page v provides an overview of the relationship among the three processors and other elements of the assembly language system.

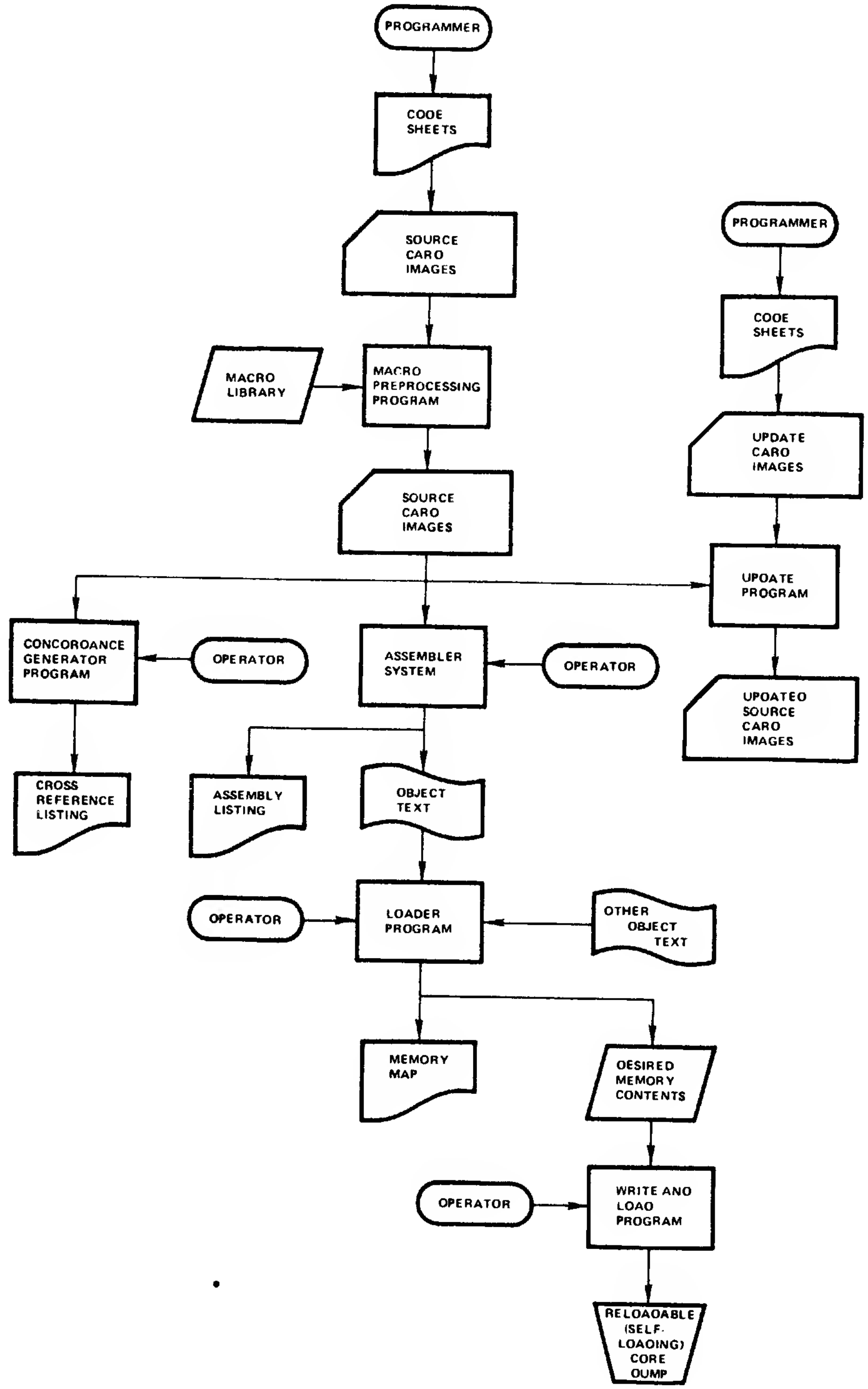
File No.: 1A23

AC94

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES



GENERAL PROGRAM FLOW

SECTION I

MAC MACRO PROCESSOR FOR DAP-16

The Macro Processor (MAC) is a single-pass processor used to expand DAP-16 Assembly Language sources containing prototype macro definitions and statements. Macro definitions can be entered from a separate library or they can be embedded in a single source. The definitions are retained in main memory for the duration of the job. As the macro prototypes are expanded according to a macro statement, the actual arguments of the macro statement replace the dummy arguments of the macro definition. The MAC output is assembly source code which is suitable for input to the DAP-16 Mod 2 Assembler.

GENERAL DESCRIPTION

The macro language provides a convenient way to generate a desired sequence of assembly language statements many times in one or more programs. The macro definition which defines the desired sequence of statements is written once near the top of the program. Then, a single statement, the macro statement, is written each time the desired code is to be generated. This simplifies program coding, reduces the chance of programming error, and ensures that standard sequences of statements are used to accomplish desired functions.

Conditional assembly can also be used with the MAC processor. In this way, statements can be coded which may or may not be assembled, depending upon conditions evaluated at assembly time. These conditions are usually tests of values, which can be defined, set, or changed during a macro expansion.

Macro Definition

Every macro definition is divided into three major components: the header (or first) statement, the body of the macro which specifies the prototype, and the end (or last) statement.

HEADER STATEMENT

The first record of every macro definition must conform to the following:

Macro Name	MAC(*)	(list of arguments used) (comments)
Field 1	Field 2 (*) - Optional	Field 3 - Optional

Field	Columns	Contents
1	1 through 4	Macro name, maximum of four characters (cannot have * or $ as first character).
2	6 through 8	The letters MAC.
	9	An asterisk if the macro expansion is to be preceded by two comment records. A space if the comment records are to be omitted.
3	12 through 72	An optional argument list, comments, and/or identification statements which are ignored by the processor.

MACRO DEFINITION BODY

The body of the macro definition (DAP-16 format assumed) consists of prototype code containing dummy variables. The dummy variables are replaced by the arguments passed in the macro statement when the statement is expanded. The expanded source code is output in the same relative position from which the macro instruction statement was read. Comment lines within the definition will be output unchanged when the macro prototype is expanded. Any occurrence of the character pair #0 will be replaced by a single digit between one and nine which represents the number of actual arguments in the macro statement (call). This digit can be less than, equal to, or greater than the number of dummy arguments, but in no case can it exceed nine.

Any occurrence of the character pairs #1 through #9 will be replaced by a copy of the corresponding actual argument when the macro is expanded. This point is illustrated in the examples which follow.

Macro statements referencing previously defined macros are allowed in the body of a macro definition. Macro names included in the argument list require additional passes to complete expansion (i.e., the MAC processor is not re-entrant). Likewise, recursive use requires an additional number of passes equal to the degree of recursiveness.

END STATEMENT

The last record of every macro definition must contain the following:

( )	ENDM	( )
Field 1	Field 2	Field 3

Field	Columns	Contents
1	1 through 5	Ignored.
2	6 through 9	The letters ENDM.
3	10 through 120	Ignored.

Macro Statement

A macro statement can call a macro any number of times after the macro has been defined by the macro definition. Each macro statement must contain the following:

(label)	Macro Name	(arg 1, arg 2. . . . arg 9) (comments)
Field 1	Field 2	Field 3

Field	Columns	Contents
1	1 through 4	Label or blank.
	5	Ignored.
2	6 through 9	Name of macro being called.
	10 and 11	Ignored.
3	12 through 72	The argument list terminated by one or more blanks. Each argument may consist of any number of characters. Any character may be used except a comma or a blank. A pound sign (#, '243) is converted into a blank during the expansion. Arguments are separated by commas. Null arguments are acceptable.
	73 through 120	Ignored.

Data Format

Source data for MAC is prepared in a manner similar to the source data for DAP-16. If the standard input routines are used, tabs will be set at columns 6, 12, 30, and 73. The backslash character can be used to compress the source tape. If MAC output is punched on paper tape using the standard output routines, backslash characters are used to tab to columns 6, 12, 30, and 73.

End-of-text records (ETX, '003 or '203 on the ASR or high-speed reader, or 11-8-6 for cards) are ignored on input and are not generated on output.

Input will be halted during a job upon detection of a MOR pseudo-operation. This feature allows the concatenation of macro libraries and other sources. It also allows the changing of input device selection.

One file per job is processed by the MAC processor. If the source is entered from paper tape or cards, the file must be terminated by a record containing a dollar sign ($, '044 or '244) in column 1. The letters END may follow in columns 2 through 4. On magnetic tape, files may also be terminated by a tape mark.

Magnetic tape input allows records of up to 120 characters. This facilitates the processing of source material with the Honeywell Series 16 FORTRAN system, which writes 120-character formatted records.

Magnetic tape output is written in 120-character records. Thus, subsequent language processors must accept this record length.

Errors

MAC recognizes five error conditions. For each of these, a message is printed on the teletypewriter in the following form:

* (error flag) record being processed

The error flags are listed in Table 1-1.

Table 1-1. MAC Error Messages

Error Flag	Description
A	More than nine arguments in the actual argument list. The request for the number of arguments (#0) is not counted.
F	Field overflow in macro expansion. An attempt has been made to store a character other than a space in or beyond columns 5, 11, 29, or 72. MAC ignores this error.
M	MAC pseudo-operation in macro definition. This record is ignored.
O	Memory overflow. There is insufficient space in the free core area to store all macro definitions. This error ends the job.
P	Formal argument not #0 through #9. A # which is not followed by a single decimal digit has been detected within a macro body. This error is not detected until the macro is expanded. MAC goes to the next field of the current record of the macro body.

SYSTEM GENERATION

The generation of a MAC system is done in a straightforward fashion. The relocatable object of MAC is loaded, usually at the default starting address of '1000. It will call the relocatable input/output supervisor, MAC-IOS, which in turn will call the off-line I/O

driver programs. The drivers for the desired devices, including the magnetic tape support package, may be loaded in any order. Finally the calls for unwanted devices must be satisfied with G$DR, which will also pass to MAC-IOS the bounds of available memory.

If G$DR is loaded in the extended desectorizing mode (EXD), MAC will be entered in the extended mode (EXA) and memory through 32K will be used. If G$DR is loaded in the normal desectorizing mode (LXD), MAC will be entered in the normal mode (DXA) and memory through 16K will be used regardless of the presence of additional memory.

The keyword is loaded at G$DR, which may be located by obtaining a core map. As loaded, the keyword contains '000022, which selects paper tape input (with parity) and output. The keyword default value may be changed at this time.

OPERATION (Keyword)

Prior to starting MAC, the source to be expanded, which must have all macro prototypes physically before they are required, must be ready, as must the output device. Set the keyword in the A register if other than the default value is used and start at the first address of MAC, normally '1000.

The source will be read until either a MOR pseudo-operation is encountered or a MAC end-of-file is detected. The latter must be either a dollar sign in column 1 or a tape mark. Any ETX records on paper tape or cards are ignored. The MOR pseudo-operation will cause the message MORE to be typed and the processor will stop for operator action. The keyword, which may be changed before restart, will be displayed in the A register.

Output may be generated immediately depending upon the input data. If paper tape is being used, MAC will punch leader and trailer.

The A register must be set with the keyword at the start of the program. If the word is zero, the default word ('000022 unless changed at system generation) will be used. If any bit is set, the A register must contain the entire desired keyword. The A register setting for the keyword is shown on the following page.

1										11	12	13	14	15	16

Process Control	Source Device	Output Device
0 - Normal output	1 - ASR	0 - No output
1 - Errors only output	2 - Paper tape reader (even parity)	1 - ASR
	3 - Card reader	2 - Paper tape punch
	4 - Paper tape reader (no parity)	3 - Card punch
	5 - Magnetic tape logical 0	4 - Line printer
	6 - Magnetic tape logical 1	5 - Magnetic tape logical 0
	7 - Magnetic tape logical 2	6 - Magnetic tape logical 1
		7 - Magnetic tape logical 2

MESSAGES

The only non-error run message is MORE, which indicates that a MOR pseudo-operation was encountered.

A MAC error message may be typed as indicated in "Errors" above. Other error messages are:

DEVICE NOT AVAILABLE

RECORD UNREADABLE (magnetic tape only)

END OF TAPE (input or output, magnetic tape only)

The end of job is indicated by:

** MACRO ERRORS

or NO MACRO ERRORS

PAPER TAPE PARITY

MAC includes support of paper tape punched with even parity. Honeywell does not supply Teletypes equipped to punch parity in the United States, but the option is common elsewhere. An error is processed within the driver program and the appropriate documentation must be consulted for advice. All output on the paper tape punch is made with level 8 marking (i.e., punched).

TERMINATION

At the end of the job, magnetic tape input is positioned after the tape mark or the dollar sign record. An output tape will have a tape mark but not an $END record. The output tape is left just beyond the tape mark.

Other output devices will be terminated with a record containing $END columns 1 through 4.

The A register will contain the keyword. A new job may be started from the halting location.

INTERNAL MAC EXPANSION PROCESSING

The input stream is scanned for macro instruction statements one record at a time. Upon detecting a macro instruction statement, two comment records are output, if the prototype indicates the selection (an asterisk in column 9 of the first prototype statement). The first optional comment record contains an asterisk in column 1, followed by blanks. The second contains an asterisk in column 1, blanks in columns 2 through 4, and a copy of the macro statement in column 5 through 120.

Following the comment records, the prototype is expanded. Each record of the macro prototype is examined and comment records within the macro definition are output unchanged. Other records are expanded field by field (i. e., columns 1 through 5, 6 through 11, 12 through 29, and 30 through 72). Any occurrence of the character pair #0 is replaced by a single digit between 1 and 9 that represents the number of actual arguments found in the input record. The character pairs #1 through #9 are replaced by their respective actual arguments listed in the macro statement (i. e., #1 is replaced by the first actual argument, #2 is replaced by the second actual argument, etc.). If the actual argument does not exist, the whole record is ignored and MAC goes to the next record of the macro prototype. All pound signs in the arguments are converted to spaces in the output record, except in the two optional comment records preceding a macro expansion.

Any noncomment records which have an operation field (columns 6 through 9) previously defined by a stored macro definition are expanded. All other records are copied unchanged.

MEMORY REQUIREMENTS

The macro prototypes are packed into core. The MAC and ENDM records require a total of seven half-words. For other records in the macro prototypes, one half-word is required for each nonspace character, one-half word is required for each sequence of spaces not at the end of a record, and one-half word additional is required for each record.

MACRO EXAMPLES

Five examples of macro applications are presented in this section. If the MAC output

is punched on paper tape, it contains backslash characters to represent tab to columns 6, 12, 30, and 73. The MAC output listings in this manual have been expanded to the correct tab position for each of reading.

Generation of Calling Sequences Using Subroutines

If a subroutine is called many times during a long program, a macro definition of the calling sequence can be written. A typical macro definition for a call to subroutine YORN is:

```
YORN MAC*   #1,#2               TEST FOR YES OR NO
     JST    YORN                YES OR NO ROUTINE
     JMP    #1                  NO
     JMP    #2                  YES
*                               INVALID CHARACTER
     ENDM                       END OF MAC YORN
     MOR
```

After the macro has been defined, a macro statement which references the macro definition can be inserted at any point in the coding:

```
     .
     .
     .
     YORN   U,V5                TEST ANSWER 5
     .
     .
     .
```

MAC expands the prototype which generates the coding shown below. The first two lines are the optional comment lines, which have been generated because the first statement of the macro definition contains an asterisk in column 9.

```
      .
      .
      .
*
*    YORN   U,V5                TEST ANSWER 5
     JST    YORN                YES OR NO ROUTINE
     JMP    U                   NO
     JMP    V5                  YES
*                               INVALID CHARACTER
      .
      .
      .
```

Use of #0 in Macro Expansion

Any occurrence of the character pair #0 will be replaced by a single digit between 0 and 9 when the macro is expanded. The number which replaces the character pair #0 represents the number of actual arguments passed from the macro statement. It can be less

than, equal to, or greater than the number of formal arguments but in no case can it exceed 9.

In the following example, the prototype code for the previous example is modified by adding the character pair #0 in three places:

```
YORN MAC*  #1,#2,#0           TEST FOR YES OR NO
     JST   YORN               YES OR NO ROUTINE
     JMP   #1                 NO,#0
     JMP   #2                 YES
*                             INVALID CHARACTER,#0
     ENDM                     END OF MAC YORN
     MOR
```

The same macro statement is used to cause the expansion shown below:

```
*
*    YORN  U,V5               TEST ANSWER 5
     JST   YORN               YES OR NO ROUTINE
     JMP   U                  NO,2
     JMP   V5                 YES
*                             INVALID CHARACTER, 0
```

Note that the first occurrence of the character pair #0 was in the dummy argument list and does not appear in the expansion. The second occurrence of #0 is in a coding line and is replaced by 2, which is the number of actual arguments. The last occurrence of #0 is in a comment line. In this case, the pound sign is taken as a space and the character 0 is printed unchanged.

Generation of Complete In-Line Coding

This type of coding may be inefficient in terms of memory used, but it is simpler than calling a subroutine and runs faster. An example containing two macro definitions is presented in Figure 1-1. The macro definitions are placed at the beginning of the program. The macro statements which reference the macro definitions are placed within the body of the program as frequently as desired.

MAC generates the coding presented in Figure 1-2 each time the prototypes are expanded.

```
        .
        .
        .
FILL  MAC*   #1,#2,#3            FROM,WITH,COUNT
      LDX    =-#3                #3 WORDS
      LDA    #2                  FILL WITH #2
      STA    #1+#3,1             STARTING WITH #1
      IRS    0                   TALLY INDEX
      JMP    *-2                 REPEAT TO END OF BUFFER
      ENDM                       END OF MAC FILL
COPY  MAC*   #1,#2,#3            FROM,TO,COUNT
      LDX    =-#3                #3 WORDS
      LDA    #1+#3,1             COPY FROM #1
      STA    #2+#3,1             TO #2
      IRS    0                   TALLY INDEX
      JMP    *-3                 REPEAT TO END OF BUFFERS
      ENDM                       END OF MAC COPY
        .
        .
        .
```

Macro Definitions

```
        .
        .
        .
      COPY   IN,OUT,36           COPY TO OUTPUT BUFFER
      FILL   IN,=A##,60          CLEAR INPUT BUFFER
        .
        .
        .
```

Macro In-Line Coding

Figure 1-1. In-Line Coding Example — Source Input

```
        .
        .
        .
*
*     COPY   IN,OUT,36           COPY TO OUTPUT BUFFER
      LDX    =-36                36 WORDS
      LDA    IN+36,1             COPY FROM IN
      STA    OUT+36,1            TO OUT
      IRS    0                   TALLY INDEX
      JMP    *-3                 REPEAT TO END OF BUFFERS
        .
        .
        .
        .
*
*     FILL   IN,=A##,60          CLEAR INPUT BUFFER
      LDX    =-60                60 WORDS
      LDA    =A                  FILL WITH =A
      STA    IN+60,1             STARTING WITH IN
      IRS    0                   TALLY INDEX
      JMP    *-2                 REPEAT TO END OF BUFFER
        .
        .
        .
```

Figure 1-2. MAC Expansion of In-Line Coding

Using Macros to Implement Interpretive Scheme

In this example, a scheme is presented which will allow the coding of a process of several steps involving complex variables. The macro definitions used in this example are presented in Figure 1-3.

With these macro definitions, it is possible to code the complex equation

Y = A * SIN (A * T1 + B * T2) * COS (B * T1-A *2)

as the series of macro statements shown in Figure 1-4.

The assembled output of MAC will be similar to the listing presented in Figure 1-5.

```
        MAC    1
        CALL   L$55                LOAD COMPLEX VARIABLE
        DAC    #1
        ENDM                       END OF MAC BLANK
+       MAC    1
        CALL   A$55                COMPLEX ADD
        DAC    #1
        ENDM                       END OF MAC +
-       MAC    1
        CALL   S$55                COMPLEX SUBTRACT
        DAC    #1
        ENDM                       END OF MAC -
X*      MAC    #1
        CALL   M$55                COMPLEX MULTIPLY
        DAC    #1
        ENDM                       END OF MAC X*
/       MAC    #1
        CALL   D$55                COMPLEX DIVIDE
        DAC    #1
        ENDM                       END OF MAC /
NEG     MAC
        CALL   N$55                COMPLEX NEGATE
        DAC    #1
        ENDM                       END OF MAC "EG
=       MAC    1
        CALL   H$55                STORE COMPLEX VARIABLE
        DAC    #1
        ENDM                       END OF MAC =
SIN     MAC
        =      W
        CALL   CSIN                COMPLEX SINE
        DAC    W
        ENDM                       END OF MAC SIN
COS     MAC
        =      W
        CALL   CCOS                COMPLEX COSINE
        DAC    W
        ENDM                       END OF MAC COS
EXP     MAC
        =      W
        CALL   CEXP                COMPLEX EXPONENTIAL
        DAC    W
        ENDM                       END OF MAC EXP
LN      MAC
        =      W
        CALL   CLOG                COMPLEX LOG
        DAC    W
        ENDM                       END OF MAC LN
SQRT    MAC
        =      W
        CALL   CSQRT               COMPLEX SQUARE ROOT
        DAC    W
        ENDM                       END OF MAC SQRT
ABS     MAC
        =      W
        CALL   CABS                COMPLEX ABSOLUTE
        DAC    W
        ENDM                       END OF MAC ABS
        MOR
```

Figure 1-3. Macro Definitions for Interpretive Scheme

```
               A                           LOAD A
         X*    T1                          A*T1
         =     W+4                         STORE A*T1
               B                           LOAD B
         X*    T2                          B*T2
         +     W+4                         A*T1+B*T2
         SIN                               SINE OF (A)
         X*    A                           SINE OF (A)*A
         =     W+4                         STORE SINE OF (A)*A
               A                           LOAD A
         X*    T2                          A*T2
         =     W+8                         STORE A*T2
               B                           LOAD B
         X*    T1                          B*T1
         -     W+8                         B*T1-A*T2
         COS                               COSINE OF (A)
         X*    W+4                         A*SIN*COS
         =     Y                           STORE Y
W        BSZ   8
Y        BSZ   4
         END
$
```

Figure 1-4. Macro Statements for Interpretive Scheme

```
00000   0 10 00000        CALL  L$55      LOAD COMPLEX VARIABLE
00001   0 000067          DAC   A
00002   0 10 00000        CALL  M$55      COMPLEX MULTIPLY
00003   0 000064          DAC   T1
00004   0 10 00000        CALL  H$55      STORE COMPLEX VARIABLE
00005   0 000054          DAC   W+4
00006   0 10 00000        CALL  L$55      LOAD COMPLEX VARIABLE
00007   0 000065          DAC   B
00010   0 10 00000        CALL  M$55      COMPLEX MULTIPLY
00011   0 000066          DAC   T2
00012   0 10 00000        CALL  A$55      COMPLEX ADD
00013   0 000054          DAC   W+4
00014   0 10 00000        CALL  H$55      STORE COMPLEX VARIABLE
00015   0 000050          DAC   W
00016   0 10 00000        CALL  CSIN      COMPLEX SINE
00017   0 000050          DAC   W
00020   0 10 00000        CALL  M$55      COMPLEX MULTIPLY
00021   0 000067          DAC   A
00022   0 10 00000        CALL  H$55      STORE COMPLEX VARIABLE
00023   0 000054          DAC   W+4
00024   0 10 00000        CALL  L$55      LOAD COMPLEX VARIABLE
00025   0 000067          DAC   A
00026   0 10 00000        CALL  M$55      COMPLEX MULTIPLY
00027   0 000066          DAC   T2
00030   0 10 00000        CALL  H$55      STORE COMPLEX VARIABLE
00031   0 000060          DAC   W+8
00032   0 10 00000        CALL  L$55      LOAD COMPLEX VARIABLE
00033   0 000065          DAC   B
00034   0 10 00000        CALL  M$55      COMPLEX MULTIPLY
00035   0 000064          DAC   T1
00036   0 10 00000        CALL  S$55      COMPLEX SUBTRACT
00037   0 000060          DAC   W+8
00040   0 10 00000        CALL  H$55      STORE COMPLEX VARIABLE
00041   0 000050          DAC   W
00042   0 10 00000        CALL  CCOS      COMPLEX COSINE
00043   0 000050          DAC   W
00044   0 10 00000        CALL  M$55      COMPLEX MULTIPLY
00045   0 000054          DAC   W+4
00046   0 10 00000        CALL  H$55      STORE COMPLEX VARIABLE
00047   0 000060          DAC   Y
00050   000000        W   BSZ   8
00060   000000        Y   BSZ   4
                          END
```

Figure 1-5. Assembler Listing for Interpretive Scheme

Macros Used With Conditional Assembly

Macro prototypes can be coded to include conditional assembly statements and the assembly process thereby directed by the value (or presence) of certain arguments. Coding using conditional pseudo-operations to call either the ASR or line printer off-line driver program is shown in Figure 1-6.

```
PRNT MAC*   #1,#2          TEST FOR OUTPUT DEVICE
     IFZ    #1-4           ZERO IF LINE PRINTER
     CALL   O$LA           CALL L.P.
     DAC    #2             BUFFER ADDRESS
     ENDC                  END OF L.P. CONDITIONAL
     IFZ    #1-1           ZERO IF ASR
     CALL   O$LL           CALL ASR
     DAC    #2             BUFFER ADDRESS
     ENDC                  END OF ASR CONDITIONAL
     IFN    #1-1           ASR TEST
     IFN    #1-4           L.P. TEST
     FAIL                  ERROR CONDITION
     ENDC                  END OF L.P. TEST
     ENDC                  END OF ASR TEST
     ENDM                  END OF MAC PRNT
     MOR
```

Figure 1-6. Macro Definition Using Conditional Pseudo-Operations

Following the macro definition, the line printer can be called using the following macro statement:

```
     •
     •
     •
     PRNT  4,DATA          1=ASR,4=LINE PRINTER
     •
     •
     •
```

When the MAC output is assembled, the program listing will contain the line printer call:

```
*
*    PRNT  4,DATA          1=ASR,4=LINE PRINTER
     CALL  O$LA            CALL L.P.
     DAC   DATA            BUFFER ADDRESS
```

Conversely, to call the ASR:

```
     •
     •
     •
     PRNT  1,DATA          1=ASR,4=LINE PRINTER
     •
     •
     •
```

results in:

```
*
*       PRNT    1,DATA              1=ASR,4=LINE PRINTER
        CALL    O$LL                CALL ASR
        DAC     DATA                BUFFER ADDRESS
```

If neither 1 nor 4 is passed as an argument, the FAIL pseudo-operation will be assembled and an error flagged:

```
*
*       PRNT    3,NODT              1=ASR,4=LINE PRINTER
        FAIL                        ERROR CONDITION
```

Trace Example Using Macros and Conditional Assembly

In this example, a macro definition to call FORTRAN IV Trace Program F$TR from an assembly language program is shown. Conditional assembly pseudo-operations are used to distinguish between a statement number (label) trace and other traces, and the call is adjusted accordingly. The macro definition used in this examp e is shown as Figure 1-7.

```
TR      MAC     TYPE,NAME
        IFZ     #1                      0=TRACE STATEMENT NO.
        CALL    F$TR
        VFD     3,#1,5,'37,8,'240  '37=HALF<,'240 PREVENTS AT SIGN
        BCI     2,#2
        ELSE
        IMA     #2
        CALL    F$TR
        VFD     3,#1
        BCI     2,#2
        IMA     #2
        ENDC
        ENDM
        MOR
```

Figure 1-7. Macro Definition for Conditional Trace Example

In the example, it was decided to trace the logical variable FLAG, the interger variable A, and the label TWO. F$TR requires the argument #1 to be 3, 1, and 0, respectively, to identify these types of variables. The macro statement TR and the variable type and name arguments were then inserted into the existing code at the points where tracing was desired. The results are shown in Figure 1-8.

```
*             TRACE EXAMPLE USING MACROS
       CF5
       REL
       CRA
LOOP   STA    FLAG
       TR     3,FLAG
       SLZ
       JMP    TWO
       STA    A
       LDA    =-4
       STA    CNT
FST    IRS    A
       TR     1,A
       IRS    CNT
       JMP    FST
       LDA    FLAG
       ERA    =1
       JMP    LOOP
TWO    CRA
       TR     0,TWO
       STA    A
       LDA    =-2
       STA    CNT
SEC    IRS    A
       TR     1,A
       IRS    CNT
       JMP    SEC
       HLT
FLAG   BSZ    1
CNT    BSZ    1
A      BSZ    1
       END
$
```

Figure 1-8. Trace Example Program

Each macro statement within the coding is expanded and the actual arguments are substituted in the MAC output coding. A portion of this coding is presented in Figure 1-9.

When the MAC output is assembled, the conditional pseudo-operations cause the inappropriate lines to be ignored. The assembly listing is presented in Figure 1-10.

Following the assembly, the program was loaded and executed. The trace shown below was printed on the Teletype. The @ signs result from some assumptions within F$TR relative to the (normally) compiler-generated call.

```
@FLAG=  F
@A=        1
@A=        2
@A=        3
@A=        4
@FLAG=  T
(TWO)
@A=        1
@A=        2
```

```
*            TRACE EXAMPLE USING MACROS
        CF5
        REL
        CRA
LOOP    STA     FLAG
        IFZ     3                    0=TRACE STATEMENT NO.
        CALL    F$TR
        VFD     3,3,5,'37,8,'240     '37=HALF<,'240 PREVENTS AT SIGN
        BCI     2,FLAG
        ELSE
        IMA     FLAG
        CALL    F$TR
        VFD     3,3
        BCI     2,FLAG
        IMA     FLAG
        ENDC
        SLZ
        JMP     TWO
        STA     A
        LDA     =-4
        STA     CNT
FST     IRS     A
        IFZ     1                    0=TRACE STATEMENT NO.
        CALL    F$TR
        VFD     3,1,5,'37,8,'240     '37=HALF<,'240 PREVENTS AT SIGN
        BCI     2,A
        ELSE
        IMA     A
        CALL    F$TR
        VFD     3,1
        BCI     2,A
        IMA     A
        ENDC
        IRS     CNT
        JMP     FST
        LDA     FLAG
        ERA     =1
        JMP     LOOP
TWO     CRA
        IFZ     0                    0=TRACE STATEMENT NO.
        CALL    F$TR
        VFD     3,0,5,'37,8,'240     '37=HALF<,'240 PREVENTS AT SIGN
        BCI     2,TWO
        ELSE
        IMA     TWO
        CALL    F$TR
        VFD     3,0
        BCI     2,TWO
        IMA     TWO
        ENDC
        STA     A
        LDA     =-2
        STA     CNT
SEC     IRS     A
        IFZ     1                    0=TRACE STAT
        CALL    F$TR
        VFD     3,1,5,'37,8
```

Figure 1-9. Typical MAC Output Code — Conditional Trace Example

#AC94

```
0001                            *          TRACE EXAMPLE USING MACROS
0002                                  CF5
0003                                  REL
0004 00000   140040                   CRA
0005 00001   0 04 00053 LOOP STA    FLAG
0011 00002   0 13 00053      IMA    FLAG
0012 00003   0 10 00000      CALL   F$TR
0013 00004   060000          VFD    3,3
0014 00005   143314          BCI    2,FLAG
     00006   140707
0015 00007   0 13 00053      IMA    FLAG
0017 00010   100100          SLZ
0018 00011   0 01 00031      JMP    TWO
0019 00012   0 04 00055      STA    A
0020 00013   0 02 00060      LDA    =-4
0021 00014   0 04 00054      STA    CNT
0022 00015   0 12 00055 FST  IRS    A
0028 00016   0 13 00055      IMA    A
0029 00017   0 10 00000      CALL   F$TR
0030 00020   020000          VFD    3,1
0031 00021   140640          BCI    2,A
     00022   120240
0032 00023   0 13 00055      IMA    A
0034 00024   0 12 00054      IRS    CNT
0035 00025   0 01 00015      JMP    FST
0036 00026   0 02 00053      LDA    FLAG
0037 00027   0 05 00057      ERA    =1
0038 00030   0 01 00001      JMP    LOOP
0039 00031   140040     TWO  CRA
0041 00032   0 10 00000      CALL   F$TR
0042 00033   0...            VFD    3,...
0043 000..
```

Figure 1-10. Typical Assembly Listing — Conditional Trace Example

SECTION II

O16-XREF CONCORDANCE

GENERAL DESCRIPTION

The O16-XREF Concordance program prepares a cross-reference listing (concordance) of all symbols within a Honeywell DAP-16 Assembly Language source program. The symbols are listed in alphanumeric order with the defining line numbers shown to the left (along with any exception flags) and the line numbers of all references to the symbol on the right. The line (record) numbers on the left are the same as the numbers shown on the left of a source listing or an assembly listing. Figure 2-1 shows an assembly listing of a source and the corresponding cross-reference. The source was written to show the features of O16-XREF.

Symbol recognition is less restrictive than the similar process in the assemblers for the same language. Symbols which are almost acceptable are detected and the programmer is made aware of them.

The four exception flags are:

M — Illegal multiple definition of symbol

N — Symbol defined but never referenced

S — Legal multiple definition of symbol

U — Undefined symbol

Flags M and U indicate lines of a finished program which are usually clearly in error. The N flag indicates a symbol which is not required by the assembler and may point out a programmer oversight. An acceptable multiple definition is indicated by an S flag.

Each reference to a symbol can have a C or J suffix to the line number to indicate a change or jump, respectively. The C indicates that the referenced instruction is a STA, DST, STX, IMA, or IRS. A J results from a JMP or JST instruction. It should be noted that the C and J suffixes are useful only when the nature of the code is known, since the actual instruction and referenced address are subject to indexing, indirect addressing, program self-modification, etc.

```
           0001                           *     XREF EXAMPLE
           0002                           *
           0003                           ABC   REL    ABCD                 NO SYMBOLS NORMALLY HERE
           0004                                 EXT    BCD                  EXTERNAL DEFINITION
           0005 00000    0 02 00000             LDA    BCD                  NO SUFFIX
           0006 00001    0 04 00000             STA    BCD                  C SUFFIX
           0007 00002    0 01 00000             JMP    BCD                  J SUFFIX
           0008                           *
           0009 00003    0 02 00016             LDA    =16                  THREE LITERALS WHICH HAVE
           0010 00004    0 02 00016             LDA    ='20                 THE SAME LITERAL VALUE, BUT
           0011 00005    0 02 00016             LDA    =$10                 A DIFFERENT SYMBOLIC VALUE
           0012 00006    0 13 00017             IMA    =AXY                 C SUFFIX FOR A LITERAL
           0013                           *
      M    0014 00007    0 000016         JKL   DAC    =16                  ONE M FLAG FROM JKL
      M    0015 00010    0 04 00007             STA    JKL                  TWO NORMAL REFERENCES
      M    0016 00011    0 02 00007             LDA    JKL
      M    0017 00012    000000           JKL   BSZ    3                    BOTH M AND N FLAGS FROM JKL
           0018                           *
      U    0019 00015    0 04 00024             STA    MNO                  U FLAG ON MNO
           0020 00016    000020                 FIN    MNO                  NO SYMBOLS NORMALLY HERE
                00017    154331

           0021                           *
           0022 00020    0 02 00023             LDA    =16                  NO POOLING SHOWN
           0023 00021    0 10 00023       PQR   JST    =16                  PQR GETS N FLAG
           0024                           *
           0025          000022           RETA  SET    *                    S FLAG ON RETA
      L    0026 00022    0 000000         183   XAC    0$LA                 ILLEGAL SYMBOL
      MV   0027          000007           RETA  SET    JKL+ABC              S FLAG
           0028                           *
      M    0029 00023    000020                 END    JKL THIS IS A COMMENT

      RETA      00024
      MNO       00025

           BCD     000000E    JKL     000007    MNO     000025    PQR     000021
           RETA    000024
           0009 WARNING OR ERROR FLAGS
           DAP-16 MOD 2    REV. B    10-20-70
```

Figure 2-1. XREF Example (Sheet 1 of 2)

```
*     XREF EXAMPLE


N      26  1B3
        3  ABC            27
U          ABCD            3
           BCD             4       5       6C      7J
M      14  JKL            15C     16      27      29
NM     17  JKL
U          MNO            19C     20
           ØSLA           26
N      23  PQR
NS     25  RETA
NS     27  RETA
           =$10           11
           ='20           10
           =16             9      14      22      23
           =AXY           12


       15  SYMBOLS
       20  REFERS
       29  RECORDS
        2  U FLAGS
        2  M FLAGS
        5  N FLAGS

  016-XREF 05 OCT 70
```

Figure 2-1 (cont). XREF Example (Sheet 2 of 2)

#AC94

Literals are treated as self-defining symbols rather than being evaluated. Therefore, the terms ='20 and =16, which the assembler may evaluate, are listed separately. The FIN pseudo-operation used to pool the literals is ignored.

SYSTEM GENERATION

The generation of an XREF system is performed in a straightforward fashion. The ORGed object of O16-XREF is loaded and this calls for the off-line I/O driver programs directly, with the exception of the Disk/Drum Operating Program (DOP) and its driver. Drivers for the desired devices, including the magnetic tape support programs, can be loaded in any order. Calls for unwanted device drivers can be satisfied with DUMY-X16.

If the Honeywell-supplied magnetic tape support package is used, the channel type and number (if DMC or DMA is to be used) must be entered. The O16-XREF program uses logical unit 1 for input (same as the assembler) and this may also require an entry. Complete information is contained in the appropriate option Programmers' Reference Manual.

When DOP is used, the linkages can be placed in sector zero either by directly entering them or by starting DOP.

The load map should be examined for a high greater than '5000. If this is the case, the contents of location GRMA ('464 in Rev. B) must be changed to the next higher address ending in octal zero. This address indicates the lowest memory usable for storing cross-reference data. Conversely, if the high address is much below '5000, the contents of GRMA can be changed to a lower address.

Locations '1000 and '1001 contain NOPs. The second location can be used for a LDA SYSP with the installation preferred keyword placed in SYSP ('506 in Rev. B) if desired.

OPERATION (Keyword)

The source to be cross-referenced must be made ready on the input device prior to starting O16-XREF. If magnetic tape is the input device (logical unit 1), it must be at the first record of the desired file. The input process is normally terminated upon detection of either an END pseudo-operation record or an end-of-file mark. If the input was from magnetic tape, the tape will be rewound at this time.

The installation-preferred keyword must be set into the A register prior to the start. Figure 2-2 is presented as a guide to keyword selection. If any field is left blank, the indicated default value for that field is assumed.

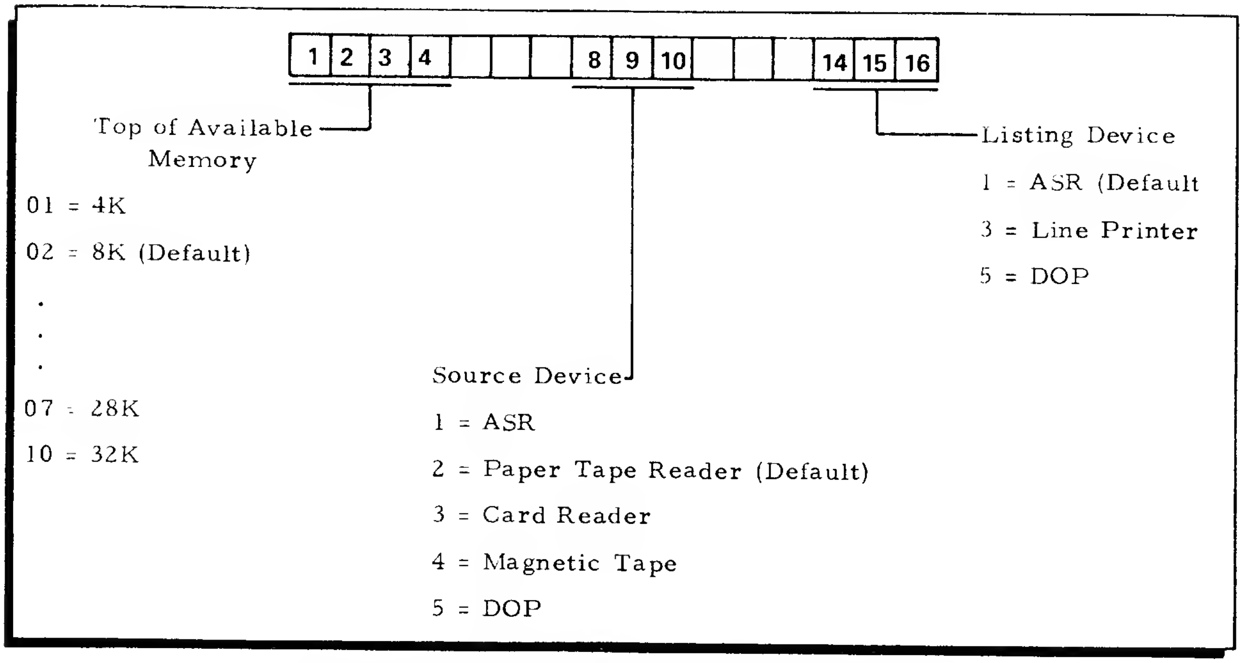


Figure 2-2. Setting of A Register for Keyword

Start the program at (P) = '1000. If the keyword is not acceptable, the message PARM ERROR is printed on the ASR and the processor halts with the keyword in the A register. The keyword must be changed and the processor restarted from either the halt address or location '1000.

At the end of the process, the processor halts and displays the keyword in the A register. Another concordance can be generated using the same or a different keyword. The concordance is listed as soon as a single input pass is complete.

MEMORY REQUIREMENTS AND OVERFLOW

On the average, a symbol is defined every five lines of code and each symbol is used five times. Each symbol requires eight words and a reference requires two words. Based on a GRMA of '5000, the source programs of various size listed in Table 2-1 can usually be cross-referenced in one operation.

If the available memory is filled before the input is complete, the cross-reference to that point is listed with flags suppressed. After the operation is complete, the message SEGMENT BOUNDARY and the cumulative counts are printed. Then, in effect, a program

restart is made which picks up the source input at the point where it left off. The final section is terminated with the message FLAGS OMITTED.

Table 2-1. XREF Memory vs Program Size

Memory	Records
4K	420
8K	1560
12K	2700
16K	3800
20K	5000
24K	6100
28K	7200
32K	8400

TERMINATION

The concordance is terminated with the record, symbol, and reference counts along with the count of M, N, and U flags. If a segmented concordance was generated, the cumulative counts of records, symbols, and references are listed along with the message FLAG OMITTED.

MISCELLANEOUS CONSIDERATIONS

The first record encountered is taken as the title line for all pages. It is also processed as a normal record. The presence of either unexpanded macro instructions or conditional assembly statements can produce warning flags which must be interpreted by the programmer.

External names are not indicated as such, but in general they may be noted by a blank-defining record field. The use of code such as

```
A$BC   XAC   A$BC
```

will obscure the external indication. References to multiply defined symbols are taken as referring to the first definition encountered. Subsequent definitions are listed, but they receive an N flag as well as an M flag.

SECTION III

SSUP SYMBOLIC SOURCE UPDATE

DESCRIPTION

SSUP and its associated Input/Output Supervisor, SSUP-IOS, form a file maintenance system for Honeywell DAP-16 Assembly Language and FORTRAN source files. Figure 3-1 presents an overview of program flow. The old master or starting source is processed against the update stream, which contains both commands and new records to be inserted. The result of the processing is either a new master, sometimes called an updated source, or both the new master and a listing.

COMMAND LANGUAGE

The update process is controlled by commands which appear in the update stream. The format of the commands is presented in Figure 3-2.

The elementary items within the square brackets may be optional. Spaces can be used freely to separate the elements, except that a blank must follow the single letter commands N and O.

The SSUP commands consist of one or four letters. The commands and their functions are presented in Table 3-1.

An F in the file/record indicator indicates that the command is to be applied to files rather than records. This mode is usually applicable to magnetic tape. When the file/record indicator is blank, the command is applied to a record.

The limit arguments, which specify record or file numbers, are decimal integers not greater than 32,767. The meaning of these arguments is described in the examples which follow. Either or both of these arguments can be absent from all commands except NSRT, OMIT, BEGN, and COPY. These four commands require at least one argument.

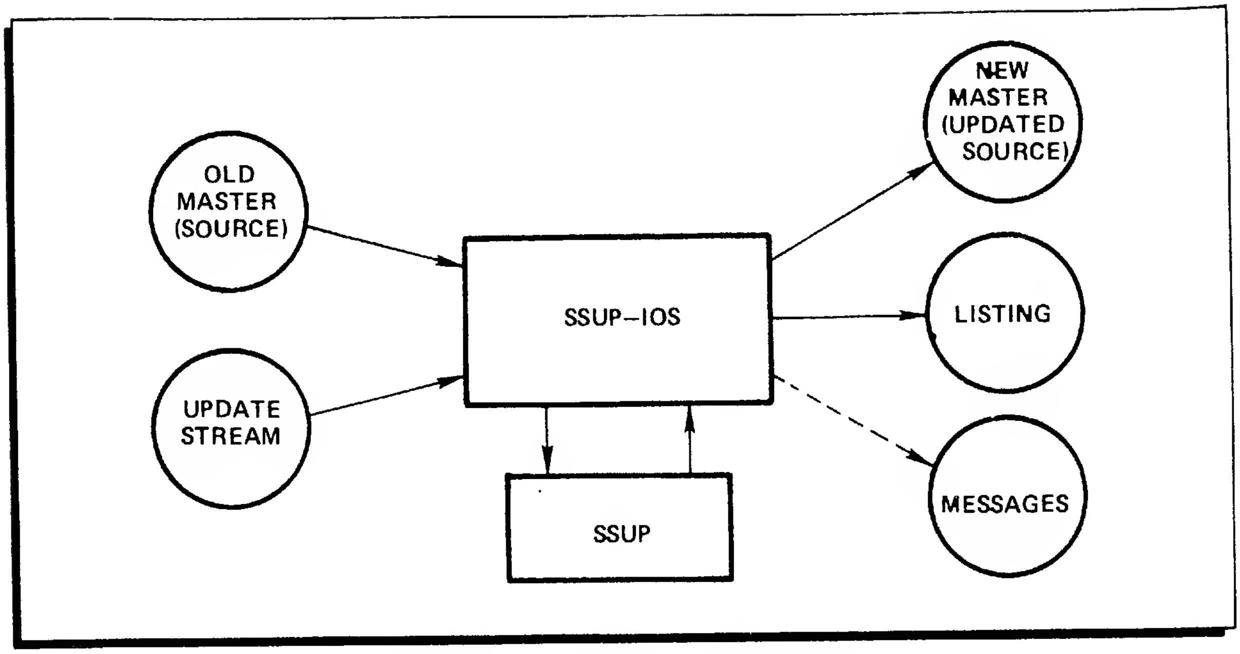


Figure 3-1. SSUP Program Flow

```
$  [Command]    [F]    [Limit 1]    [, Limit 2]    [*Comment]
```

[F]: File/Record Indicator

F = File

Blank = Record

$: Command/New Data Indicator[1]

$ = Command

Other = New Data

[1] $0 is treated as a data card since it has a meaning to the FORTRAN compiler.

Figure 3-2. SSUP Command Format

An asterisk anywhere in the command causes the remainder of the line to be treated as a comment. Hence, the remainder of the line is ignored for processing, but it is listed if commands are being listed.

A command which contains only a comment is acceptable.

Table 3-1. SSUP Commands

Command	Function
NSRT or N	Insert new records.
OMIT or O	Omit old records or omit old records and insert new records.
LIST	List all records that come from old master plus all corrections that come from update source.
NLST	No records are listed. SSUP is initially in NLST mode.
BEGN	Position new master (used with magnetic tape only).
COPY	Copy old master on new master.
NCPY	Not generally useful: turns copy indicator off and nothing is written on new master.
HALT	Stop all I/O operations. Wait for operator action.
RSET	Reset input counters to record zero, file one (useful for concatenating old sources onto one new source).
WEOF	Place end-of-file mark in new master.
DONE	Place end-of-file mark in new master and terminate SSUP process.
DOLR	Change command indicator (originally $) to next nonblank character. Useful for updating master files.

SUMMARY EXAMPLE

The example in Figure 3-3 is a typical update stream which uses the more common commands and functions. It is correct as shown for the old and new masters on magnetic tape and the update stream on cards. The flow is generally correct regardless of the media.

In general, there are several combinations of commands and limits which will accomplish any task. The example presented in Figure 3-3, with reasonable extensions by the reader, is a straightforward approach which can accomplish all ordinary update tasks.

Operation	Command	Functions	Comments
Position Output Tape	$BEGNF m $BEGN n	Advance output (new master) tape past m-1 file marks. Advance output tape past n-1 records,	Tape is now in position to write n-th record on m-th file of output tape.
Position Input Tape	$OF, p $COPYF, q	Skip files from current position to beyond file p. Copy from current position through and including file q.	New tape contains some number (between none and all) of files from old master.
Perform Update	$ N r (new record) . . . (new record) $ O t [(new record) . . . (new record)] $ O u, v [(new record) . . . (new record)]	Copy old master from current position up to and including (r-1)th record. Insert following records until the next command is found. Copy old master from current position up to and including (t-1)th record. Insert following records, if any. Copy old master from current position up to and including (u-1)th record. Skip records u through v. Insert following records, if any.	
Copy Remainder of Input Tape Terminate the Process[1]	$COPYF, w $ WEOF $ DONE	Copy old master from current position (record v+1) through [illegible]. Write end-of-file mark. Write end-of-file mark and indicate job is complete.	

[1]Two successive end-of-file marks (i.e., a blank file) customarily indicate the end of data on a magnetic tape.

Figure 3-3. SSUP Summary Example

LISTING

Two types of listings, plus a no-listing option, can be obtained from SSUP. One listing is a transaction listing which is the default condition. All commands and comments are listed, along with the results of omit and insert actions. The records inserted are shown with a plus sign and the record number they acquired in the new master. The records omitted are shown with a minus sign and the record number they had in the old master. The transaction listing may be suppressed either by setting sense switch 4 or by including the LIST command.

The LIST command overrides the transaction listing and causes the second type of listing to be generated. In this type, a line is output for every record read from the old master or the update stream. If there is no update, the resultant list is a source listing with the decimal record numbers printed on the left side. If no limits are stated, all records will be listed. If only [, limit 2] is stated, all records from the current position through record "limit 2" will be listed.

NSLT turns the LIST option off and restarts the transaction listing.

Listings produced on the teletype are truncated after card image column 72. The tab format source is expanded to the conventional columns before being listed (i.e., tabbed).

MESSAGES

There are two SSUP error messages, NONVALID COMMAND and LIMIT ERROR. An error message is followed by the current input (usually a command), the current file, and record counts. A reminder is also printed that informs the operator he may override the error or terminate the process. The override is accomplished by pushing START with sense switch 3 reset. The update stream will be ignored until the next command record is detected. If sense switch 3 is set, the effect is the same as a DONE command. These error messages are shown in the examples.

There is one I/O error message possible if the old master is being read from magnetic tape. A nonrecoverable read error will cause the message PARITY ERROR and the file and record counts to be printed. Following the printout, the computer halts. If the A register is cleared, another read will be attempted when START is pushed. If the A register is not cleared, the record will be accepted as read. Sense switch 2 will suppress the error message when it is set, but not the halt (with all bits set in the A register).

Unexpected end-of-file marks or end-of-tape being sensed will cause a halt without an error message.

Other messages are BREAK POINT HALT and END OF JOB. The BREAK POINT HALT is output when a HALT command was encountered, presumably for the operator to change the input stream.

RESEQUENCE OPTION (Keyword)

If the output is magnetic tape, SSUP can resequence card image columns 77 through 80 by either 1's or 10's depending upon the keyword in the A register at the start. If resequencing is selected, the ID field (columns 73 through 76) is taken from the first record to be written on the new master. The A register keywords for the various options are:

'000000 - Resequence by 1's

'100000 - Resequence by 10's

'040000 - Copy input verbatim

SENSE SWITCH OPTIONS

The functions of the sense switches are presented in Table 3-2.

Table 3-2. SSUP Functions of Sense Switches

Switch	Condition	Function
1	Reset Set	DAP-16 format FORTRAN format
2	Reset Set	Normal Suppress parity error message
3	Reset Set	Continue process on error restart Terminate process on error restart
4	Reset Set	Normal Suppress transaction listing

SOURCE FORMAT

If paper tape or the Teletype is used, the format must be either the correct tab form for the specific language or in acceptable columns. FORTRAN tab settings are at columns 6, 7, and 73. DAP-16 tabs are at columns 6, 12, and 30. Lines are opened with a line feed and closed with a carriage return.

Magnetic tape must have 80-character records.

OPERATION

The operator must load an SSUP system configured for the desired devices, since SSUP cannot be configured at run time. If magnetic tape is to be used, the old master must be mounted on logical unit 1 and the new master on logical unit 2. This is consistent with the assembler and compiler assignments.

The A register must be set with the keyword, and the computer started at location '1000. Successive jobs must have the keyword restored and started at location '1000.

If the output is being punched on paper tape, the operator must manually punch leading and trailing blanks and turn off the punch at the end of the job.

SYSTEM GENERATION

An SSUP system must be generated for each combination of I/O devices desired. The allowed combinations and required device drivers are:

Old Master	-	Magnetic Tape (logical unit 1)	I$MA
	-	Paper Tape Reader	I$PA
New Master	-	Magnetic Tape (logical unit 2)	O$MA
	-	Paper Tape Punch	O$PA
Update	-	Card Reader	I$CA
	-	Teletype	I$AA
	-	Paper Tape Reader	I$PA
Listing	-	Teletype	O$LL
	-	Line Printer	O$LA
Messages	-	Teletype	O$LL

The use of the Teletype as both the update master and the listing device is acceptable. The order of loading must not conflict with the above if the desired device stream attachments are to be made.

If other combinations of I/O devices are desired it is possible to attach them by writing a driver which appears to SSUP-IOS as an allowable device. This is not supported by Honeywell.

SSUP will operate in a 4K memory computer, but at least 8K is required to generate a system.

Load the ORGed SSUP and SSUP-IOS. Then select and load (consistently with the previously noted order) the desired device driver and support programs for the four streams. If the Teletype I/O library which includes O$LL was not loaded, do so at this point for the message output. Progress should be checked at this point by generating a memory map and observing that the correct calls are satisfied. Satisfy the remaining calls by loading SSUP-RDS, the associated dummy routine set.

If magnetic tape is to be used, the type of I/O channel (DMA or DMC) and the channel number along with the logical-dialed number relationship must be entered. Refer to the Programmers' Reference Manual of the appropriate option for further instructions.

EXAMPLE OF PAPER TAPE UPDATE

The following file was punched with a Teletype to demonstrate the use of SSUP. It anticipates DAP format (i.e., tab placement). All records start with a line feed and close with a carriage return. The use of an X-OFF and rubout is not required, since it will only be read by a paper tape reader. A nonprinting record following the printing text consists of a line feed, ETX, and carriage return. The ETX, which is punched by simultaneous CTRL and C keys, is recognized as an end-of-file.

```
ORG\REC\ONE
ORG\REC\TWO
ORG\REC\THREE
ORG\REC\FOUR
ORG\REC\FIVE
```

The first step in most source updates is to obtain a list of the file with record numbers. In this case a copy of the file after corrections that were made during the original punching is also obtained. Note that the Teletype shows an echo of the commands.

```
$LIST

        $LIST

$COPYFI

        $COPYFI

  00001 ORG   REC    ONE
  00002 ORG   REC    TWO
  00003 ORG   REC    THREE
  00004 ORG   REC    FOUR
  00005 ORG   REC    FIVE
$DONE

        $DONE


 END OF JOB
```

The programmer decides that records 2 and 3 are to be replaced and a new record is to be inserted between records 4 and 5. He elects to list the transactions by leaving sense switch 4 reset.

```
$ 0  0002,3  *  FREE FORM - "O" AND "OMIT" ARE SYNONYMS

        $ 0  0002,3  *  FREE FORM - "O" AND "OMIT" ARE SYNONYMS

 -00002 ORG  REC   TWO
 -00003 ORG  REC   THREE
NEW RECORD AAA
 +00002 NEW RECORD AAA
$DOLR .  * CHANGE COMMAND INDICATOR

        $DOLR .  * CHANGE COMMAND INDICATOR

.N 5

        .N 5

$NEW RECORD BBB HAS A "$" AS THE FIRST CHARACTER
 +00004 $NEW RECORD BBB HAS A "$" AS THE FIRST CHARACTER
.COPYFI

        .COPYFI

.DONE

        .DONE

 END OF JOB
```

Since the new record starts with a control character, the command indicator in use was changed from a dollar sign to a period.

As a final step the programmer listed the results. He chose not to copy the tape and because of this the listing shows each record as being deleted. In typing the commands, two errors were made to show the error messages.

```
$LIST

        $LIST

$OMIRF1

 NON-VALID COMMAND

$OMIRF1

 FILE   00001 ,RECORD   00001

SSW3 SET TO TERMINATE, RESET TO PROCESS NEXT COMMAND  DEPRESS START.

$OMITF2,1

        $OMITF2,1


 LIMIT ERROR.

$OMITF2,1

 FILE   00001 ,RECORD   00001

SSW3 SET TO TERMINATE, RESET TO PROCESS NEXT COMMAND  DEPRESS START.

$OMITF1

        $OMITF1

 -00001 ORG  REC   ONE
 -00002 NEW RECORD AAA
 -00003 ORG  REC   FOUR
 -00004 $NEW RECORD BBB HAS A "$" AS THE FIRST CHARACTER
 -00005 ORG  REC   FIVE
$NCPY

        $NCPY

$DONE

        $DONE


 END OF JOB
```

EXAMPLE OF MAGNETIC TAPE UPDATE

The first step to this example was to create three files by inserting records before the first record of a non-existent file. The cards used are shown as Figure 3-4.

The programmer decided to update his source files by eliminating the first, changing the second, and keeping the third intact. The original (or old master) remains intact. The new master contains entirely new data.

Figures 3-5 and 3-6 are the transaction listing and a listing of the new master.

```
$       N         1                          *CREATE FILE ØNE
FILE    ØNE       REC ØNE                                           FIL1
FILE    ØNE       REC TWO
$       WEOF                                 *END ØF FILE ØNE
$       N         1                          *CREATE FILE TWØ
FILE    TWØ       ØRIG REC ØNE                                      FIL2
FILE    TWØ       ØRIG REC TWØ
FILE    TWØ       ØRIG REC THREE
FILE    TWØ       ØRIG REC FØUR
FILE    TWØ       ØRIG REC FIVE
$       WEOF                                 *END ØF FILE TWØ
$       N         1                          *CREATE FILE THREE
FILE    THREE     REC ØNE                                           FIL3
FILE    THREE     REC TWO
$       WEOF                                 *END OF FILE THREE
$       WEOF                                 *END OF VOLUME
$       DONE
```

Figure 3-4. SSUP Example of Source for Magnetic Tape Update

```
          $    LIST

          $    OMITF ,1                *  SKIP A FILE

  -00001 FILE ONE    REC ONE                                                FIL10001
  -00002 FILE ONE    REC TWO                                                FIL10001
  -00003      END                                                           FIL10001

          $    O     2,3

   00001 FILE TWO    ORIG REC ONE                                           FIL20001
  -00002 FILE TWO    ORIG REC TWO                                           FIL20002
  -00003 FILE TWO    ORIG REC THREE                                         FIL20002
  +00002 NEW RECORD AAA                                                     FIL20002

          $    DOLR  .

          .    N     5

   00003 FILE TWO    ORIG REC FOUR                                          FIL20003
  +00004 $NEW RECORD BBB STARTS WITH A DOLLAR SIGN                          FIL20004

          .    COPYF ,3                *  REST OF TWO AND THREE

   00005 FILE TWO    ORIG REC FIVE                                          FIL20005
   00006      END                                                           FIL20006
   00001 FILE THREE REC ONE                                                 FIL30001
   00002 FILE THREE REC TWO                                                 FIL30002
   00003      END                                                           FIL30003

          .    WEOF                    *  END OF VOLUME

          .    DONE
```

Figure 3-5. SSUP Example of Full Listing During Update

#AC94

```
FILE TWO   ORIG REC ONE                                          FIL20001



FILE TWO   ORIG REC ONE                                          FIL20001
NEW RECORD AAA                                                   FIL20002
FILE TWO   ORIG REC FOUR                                         FIL20003
$NEW RECORD BBB STARTS WITH A DOLLAR SIGN                        FIL20004
FILE TWO   ORIG REC FIVE                                         FIL20005
     END                                                         FIL20006



FILE THREE REC ONE                                               FIL30001



FILE THREE REC ONE                                               FIL30001
FILE THREE REC TWO                                               FIL30002
     END                                                         FIL30003
```

Figure 3-6. SSUP Example of New Master

#AC94

COMPUTER GENERATED INDEX

COMPUTER GENERATED INDEX

AC94

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form*

TITLE: SERIES 16 O16-XREF, SSUP, AND MAC SOURCE LANGUAGE PROCESSORS

ORDER No.: AC94, REV. 0

DATED: JULY 1971

ERRORS IN PUBLICATION:

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION:

(Please Print)

FROM: NAME ______________________ DATE: ____________

COMPANY ______________________

TITLE ______________________

____

____

ONG LINE

C

*Your comments will be promptly investigated by appropriate technical personnel, action will be taken as required, and you will receive a written reply. If you do not require a written reply, please check here. ☐

CUT ALONG LINE

FIRST CLASS
PERMIT NO. 39531
WELLESLEY HILLS,
MASS. 02181

Business Reply Mail
Postage Stamp Not Necessary if Mailed in the United States

POSTAGE WILL BE PAID BY:

HONEYWELL INFORMATION SYSTEMS
60 WALNUT STREET
WELLESLEY HILLS, MASS. 02181

ATTN: PUBLICATIONS, MS 050

Honeywell

The Other Computer Company:

Honeywell

HONEYWELL INFORMATION SYSTEMS

10519
1474
Printed in U.S.A.

In the U.S.A.: 200 Smith Street, MS 061, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario

AC94, Rev. 0